Spring 5-23-2017

Evaluation of the Efficiency of an ARM-based Beowulf Cluster versus Traditional Desktop Computing for High Performance Computing

Nicholas Addiego
University of San Diego

Evaluation of the Efficiency of an ARM-based Beowulf Cluster versus Traditional Desktop Computing for High Performance Computing

_____

A Thesis

Presented to

The Faculty and the Honors Program

Of the University of San Diego

_____

By

Nicholas Addiego

Mechanical Engineering

2017

Table of Contents

Abstract

In the realm of scientific computing, it has become increasingly important to focus on results driven growth (Kamil, Shalf and Storhmaier). Doing this enables researchers to continue building the rapidly expanding area of scientific discovery. However, with the growth comes a cost of the amount of resources consumed to accomplish these results. Large supercomputers are consuming power at a rate roughly fourteen thousand times that of a traditional American household (U.S. Energy Information Administration). Parallel to this, public consumers have been driving the mobile industry and the research behind it. The need to have faster and faster mobile devices that can last all day long on a single battery charge has driven the development of Advanced Reduced Instruction Set processors developed by ARM. These processors are built to perform efficiently while still maintaining the ability to perform the necessary calculations. This study looked at combining these two parallel realms and analyzing the overall efficiency and energy consumption of multiple ARM processors as compared to a traditional desktop computer. The results showed that the ARM processors were less efficient roughly by an order of two when compared to the slowest possible trial on the desktop. Several variables played a significant role in these results including the limitation on network speed and bandwidth, idle energy consumption, and individual power regulators.

Introduction

The necessity of supercomputers has grown significantly as the problems they are used to compute have grown increasingly over the years. The issue that comes with this is the focus on the results and not on the amount of resources consumed. The Tianhe-2 Supercomputer in

Guangzhou, China consumes 17,808,000W of power, nearly $3,000 dollars of electricity per hour (Meuer, Strohmaier and Dongarra). To put this into perspective, a traditional 60W lightbulb will have to operate continuously for 34 years to equate to the same amount of energy consumed by the Tianhe-2 over one hour. Without focusing on increasing the sustainability of the computational processes that we use in society, the current trend of consumption in supercomputers will no longer be economical or environmentally viable. This project aimed at looking at how ARM based clusters would compare against traditional desktops when used for large computations in terms of the overall energy and power consumption, the overall efficiency, and the total amount of time it takes to complete the computations.

Supercomputers

The necessity of supercomputers originally arose when certain equations were developed describing physical phenomena that could not feasibly be solved without the assistance of computers. These equations, deemed The Grand Challenge Equations by the San Diego Supercomputer Center (see Figure 1), were what originally drove the necessity of the supercomputer (San Diego Supercomputer Center). Most of these equations cannot be solved using symbolic math and must be solved numerically.

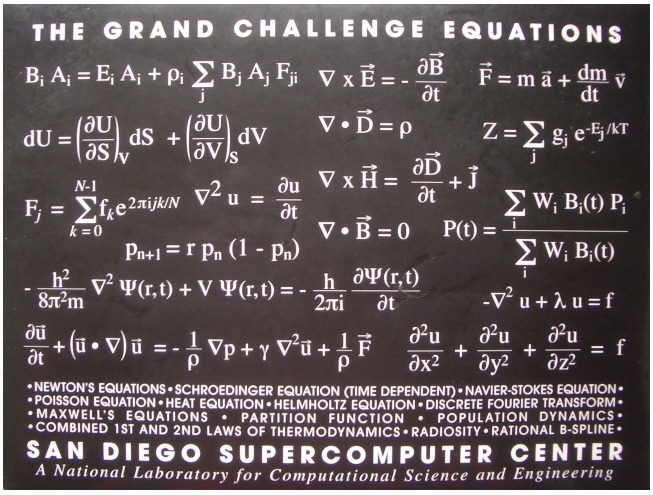

Figure 1: The Grand Challenge Equations. Image from Duncan Hull; 13 January 2007; Web; 15 April 2017

<https://www.flickr.com/photos/dullhunk/359634390/in/photostream/>

Computational Fluid Dynamics

Of these equations, one in particular was of interest to this project: The Navier-Stokes Equation (see Equation 1). The Navier-Stokes equation is a partial differential equation that represents the motion of fluids in space and how they relate to the forces at play with that motion. In essence, the Navier-Stokes Equation represents Newton's Second Law of Motion for fluid applications. The main issue is this equation does not have a traditional mathematical solution and therefore must be solved using experimental results to understand the equation better or by using creative approaches that approximate the individual problematic terms. These creative solutions are what is known as numerical approximations.

$$\frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \nabla)\vec{u} = -\frac{1}{\rho}\nabla p + \gamma \nabla^2 \vec{u} + \frac{1}{\rho}\vec{F}$$

Equation 1: Navier-Stokes Equation

Numerical Approximations

Numerical approximations are used to approximate the partial differential terms inside the Navier-Stokes equation. These terms are the hardest to account for when finding a solution to the equation. There are two different ways to approach the individual partial derivative terms. The first approach deals with the partial differential terms that are taken with respect to time. The most common method to find these derivatives is to approximate the slope of the function in question. The Runge-Kutta 4^{th} Order Numerical Approximation Method is an approach that uses multiple values in front of and behind the variable of interest to estimate the instantaneous value of the slope. This approximation is done by transforming the partial differential equation into a set of ordinary differential equations with set initial conditions that are much easier to solve. The higher the order of the approximation allows for higher accuracy and representations of the differential equations. However, with the higher accuracy, there is increased computational time as there are more values necessary to compute the slope.

The other approach is used to find the partial differential terms with respect to a direction. The Fourier Transform allows for a function that is represented in terms of physical quantities and transforms them into quantities represented by waveforms. These transforms allow for complex partial differential relations to be calculated with simple linear math. The partial differential of a variable with respect to one direction can be calculated by multiplying the result of a Fast Forward Fourier Transform (FFT) by the imaginary unit and a particular wave number. This result can then be sent through the Inverse Fast Forward Fourier Transform (IFFT) to achieve the partial differential term with respect to the direction used in the wave number

(see Figure 2). It is important to note that the Fourier approach to calculating these terms is costly with respect the amount of resources consumed. While they can be calculated using other approaches, the result is not nearly as accurate as the Fourier Method.

$$U \xrightarrow{FFT} \widehat{U} \rightarrow i \cdot k_x \cdot \widehat{U} \xrightarrow{IFFT} \frac{\partial U}{\partial x}$$

Figure 2: Fourier Transformation Scheme

Parallel Computing

In the computing world, there are two main approaches to how data can be handled. In most typical software applications, all the necessary commands are executed one after the other where the next command cannot execute until the previous one has completed. This approach of allowing only one instruction to be handled at a time is called serial computing. The other method is called parallel computing. Parallel computing involves taking the same commands and breaking them into smaller components where the different parts can be executed simultaneously. Parallel computing is taking a serial computation and dividing it into several smaller serial problems that execute in tandem. This method allows for computations that are independent of one another to be calculated at the same time, which significantly reduced the overall computational time required.

Beowulf Approach

The three main methods of implementing a parallel computing structure are a Fail-over cluster, a load-balancing cluster, and a high-performance cluster (Buytaert). The Fail-over cluster involves two separate computing entities that share a common network, allowing them

to communicate with each other. An outside resource, often called the 'heartbeat connection,' monitors when the computers are not in use and assigns additional tasks to them if they are idle.

The second type of cluster is called a load-balancing cluster. This approach is similar to the method that the fail-over cluster uses, but adds the additional functionality of load distribution. When a request or process is added to the que, the cluster looks at all its resources, determines which is the least busy, and assign the task to that node. The most common type of load-balancing clusters is for web-servers. When an internet user enters a URL to a website, the server, or cluster, that is assigned to that web domain will assign your visit to a specific node in the cluster that will interface your computer for the duration of the visit.

The third type of cluster is what is known as the High Performance Computing Cluster, often called a HPC. HPC's are most commonly seen in the scientific community because they are specifically used when large amounts of resources are dedicated to a single task. Most of the supercomputers that are mentioned in the news are HPC's. One method of assembling a HPC is by combining many outdated or unused computers to get a cluster that can outperform an up to date desktop. This method of combining computers is called a Beowulf cluster.

The initial reason that brought about the Beowulf cluster was that most traditional supercomputers were well out of the price range of smaller institutions. Donald Becker and Thomas Sterling, both from The Massachusetts Institute of Technology, conceptualized being able to provide Commodity Off the Shelf (COTS) supercomputers. COTS systems approach the dichotomy of price and computational ability by networking and combining cheap computers

that are traditionally marketed towards the home user. By sacrificing some portion of the total theoretical computational power, COTS systems are able to provide increased computational power while significantly reducing the overall cost.

Rankings

In the supercomputing world, the most important list to be on is the Top 500 (TOP500). Since 1993, TOP500 has maintained and updated the list of the 500 most powerful computers in the world. These computers are rated on the number of floating-point calculations per second, or FLOPS, they can perform. It is not uncommon to see machines that are able to perform on the order of a gigaflop, or one billion calculations per second. The one problem with these computers is they do not care about the amount of power that they consume.

Computers have been following a trend ever since their inception. Every two years, the performance of processors had been doubling as more advance processes allow scientists to get more transistors onto the processor. This trend was first noticed by Gordon Moore and was eventually named Moore's Law. Since 2006, Intel has been changing its outlook on Moore's Law to also include a more efficient model of each upgraded processor (Greene). Since laptops and mobile phones have become more abundant, the value of efficiency has only grown.

This shift has also been seen in the supercomputing world. Like the Top500, the Green 500 is a list that ranks supercomputers based on their performance. However, the clusters in the Green 500 are ranked based on the numbers of FLOPS they can do per Watt of power. The value of each cluster is how many calculations it can do per unit of energy rather than how

much it can do regardless of energy. The most efficient cluster on the Green500 list, a NVIDIA cluster, is able to perform roughly three times the amount of computations per unit of energy when compared to the Tianhe-2 supercomputer (Scogland and Feng).

ARM Processors

With the rise of mobile phones and other handheld devices, it became increasingly more important to reduce the overall power consumption of processors. As a result, the Acorn Computer Group researched and developed the first ARM processor coined the Acorn Reduced Instruction Set Machine. The Reduced Instruction Set Machines (RISC) were modified in the early 1990s in response to the emergence of the Personal Digital Assistant (PDA). This change aimed to reduce the amount of power dissipation that occurred when the clock cycles of the processor went unused. This allowed for the battery to last significantly longer, which was necessary in handheld applications. With monetary assistance from Apple Inc., the Acorn Computer Group became a new semiconductor company aimed at creating a new microprocessor standard. ARM based chips have continued to evolve crating faster chips that still prioritize the reduced overall power consumption (Levy).

Prior Work

In total, very little research has been conducted on this topic because it has only recently become relevant in the scientific and engineering community. ARM processor speeds are still very limited in comparison to modern AMD and Intel processors. Roughly, a modern ARM processor is at the same level of a basic desktop processor of ten years ago. However, one study from Nikilesh Balakrishnan of the University of Edinburgh also looked at the current

viability of ARM based clusters. Balakrishnan used a software program that does a known computation and analyzes the power it takes to complete in watts. This analysis is the same process that is used for larger TOP500 level clusters. Balakrishnan only used this data metric as a comparison between his cluster and other clusters. His work does not consider any comparison between computers that many other individuals would use. In addition, the cluster that he built is very outdated for 2012, even for ARM based clusters.

A later study from Boise State University used more modern Raspberry Pi ARM Processors and measured the relative speed up with the addition of nodes as well as the total amount of energy that the cluster would consume in idle, running, and overclocked conditions. The overall power consumption was measured at the wall and divided into the individual components that were drawing current. The study found that of the total power draw of 167.2 watts, 61.2% of that power was used by the compute nodes in their idle state (Kiepert). This value seems very large compared to what is traditionally thought of for electronics in an idle state. The value indicated that looking at the amount of energy consumed in the idle state would be a key data metric to analyze.

Hardware

The initial project focused on building a physical cluster that was able to contain all of the components necessary for it to function effectively. To this effect, it was necessary to analyze several key variables including overall current draw, total heat dissipation, and the minimum about of space where the cluster could still be functional.

Odroid XU3 and XU4

To analyze the efficiency of the ARM processor, it is essential to obtain a commercially available computer that supports the processor with the necessary hardware for computations. Such necessary hardware includes Random Access Memory (RAM), Ethernet and Universal Serial Bus (USB) ports, and hardware controllers to support the USB and Ethernet interfaces. For this study, a combination of HardKernel based Odroid boards were used to construct the Beowulf cluster (HardKernel co., Ltd.). An Odroid XU4 was used for the master node, a node used to control all other nodes, because of the availability of multiple USB3.0 ports and a Gigabit Ethernet interface. The individual compute nodes were composed of 32 Odroid XU3 Lite boards (see Figure 3). The XU3s were used because they were being retired from the Shiley-Marcos School of Engineering. The XU3 and the XU4 are both based off a 32-bit architecture that allows them to run off the same software and makes them fully compatible with each other.

By comparing the physical capabilities of the Odroid XU boards against a similar product, the Raspberry Pi 2 Model B, a better understanding of how capable the XU line is can be obtained (Raspberry Pi Foundation). The two most critical components that allow for any computer to run faster computations are the clock speed of the processors and the amount and speed of the RAM. Increased clock speed allows for more values to be computed per unit of time. The Odroid XU3 Lite, the slowest board present in the cluster has a clock speed of 1.8 GHz as compared to the Raspberry Pi's clock speed of 0.9 GHz. In terms of raw computational ability, the XU3 Lite has twice the computational ability of the Raspberry Pi (HardKernel). The

other physical quantity that can be used to compare the boards is the amount and speed of RAM present. RAM is a type of memory that can be quickly accessed by the processor, which allows for related computations to be performed faster than if they relied on virtual memory, data that is written to a hard drive. The XU3 Lite has two GB of RAM while the Raspberry Pi 2 only has one GB. Again, the XU3 Lite has nearly twice the capability to perform the computation compared to the competition (Raspberry Pi). Overall, the Odroid platform is the most capable platform for comparing the onboard ARM processors because they have the most computational ability while still using very little power overall.

Figure 3: Odroid XU3 Lite. Image from HardKernel; Web; 10 May 2017

< https://dn.odroid.com/homebackup/XU3LiteMain.jpg>

Mechanical Design

For the cluster to operate effectively and add mobility, it was necessary to build a temporary enclosure to house all the individual components in an organized fashion that still allowed for effecting cable routing and management as well as for air flow to ensure that each of the components was kept cool. In order to achieve this, all of the components would be

affixed to a series of plywood boards. The individual boards would then be threaded onto a steel rod and pinned in place by a series of washers. To accomplish this, all the components were dimensioned and recreated inside of Autodesk Inventor, a commercially available computer aided design platform (see Figure 4). From there, they were laid out on a digital representation of the plywood, which enabled the ability to plan for cable pathing, cable management, and enough space between them to allow for air flow. The plywood boards were then physically cut using a laser cutter and the individual components were placed and pinned in place. For the large equipment, including switches, a piece of networking equipment used to direct individual communications between individual nodes, and the external hard drive, a series of interconnect zip ties were used to mate the components to the board and keep them from moving. For the individual compute nodes, it was essential to offset the hardware from the board itself. To do this, a threaded spacer was screwed into the wood and the individual boards were then attached to these spacers using more screws. Doing this allowed for airflow to pass on all sides of the board and insulated the electrical connections from physical interactions (see Figure 5 and Figure 6).

The current design has certain limitations that should be noted for their impact on other components of the cluster. As all the components are mounted on one side of the plywood boards, a moment caused each of the boards to lean into other boards. This combined force contributed a large deflection of the top portions of the boards and caused them to constantly be at an angle. The other problem that occurred with this design was the shear force that occurred when the cluster was moved. When lifted, each of the boards would shear vertically

dependent on the weight of each individual board. If lifted at the wrong place, the material supporting the heavier boards could fail and cause that board to fall.

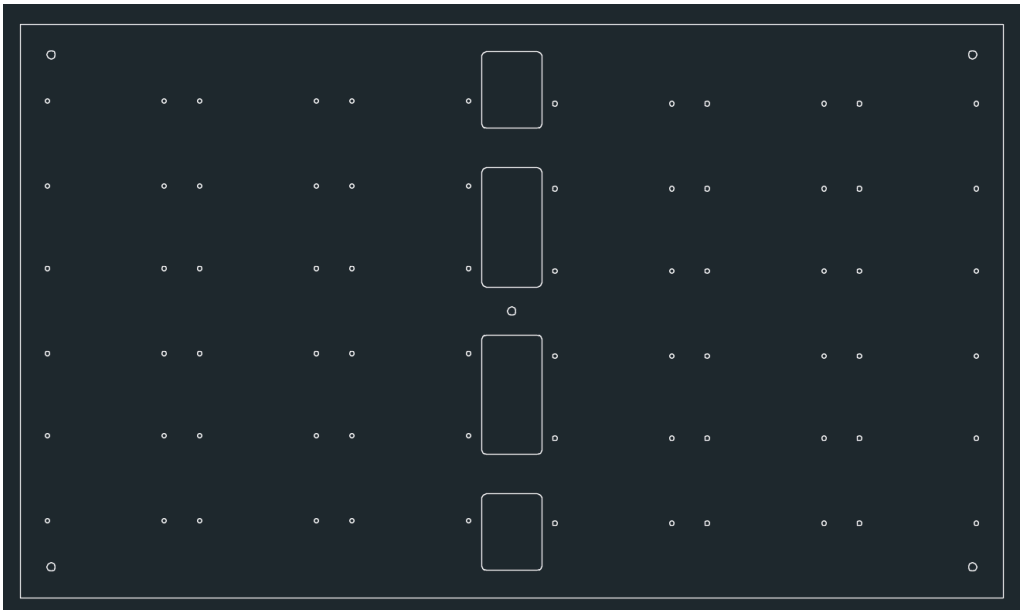

Figure 4: Computer Aided Design Layout

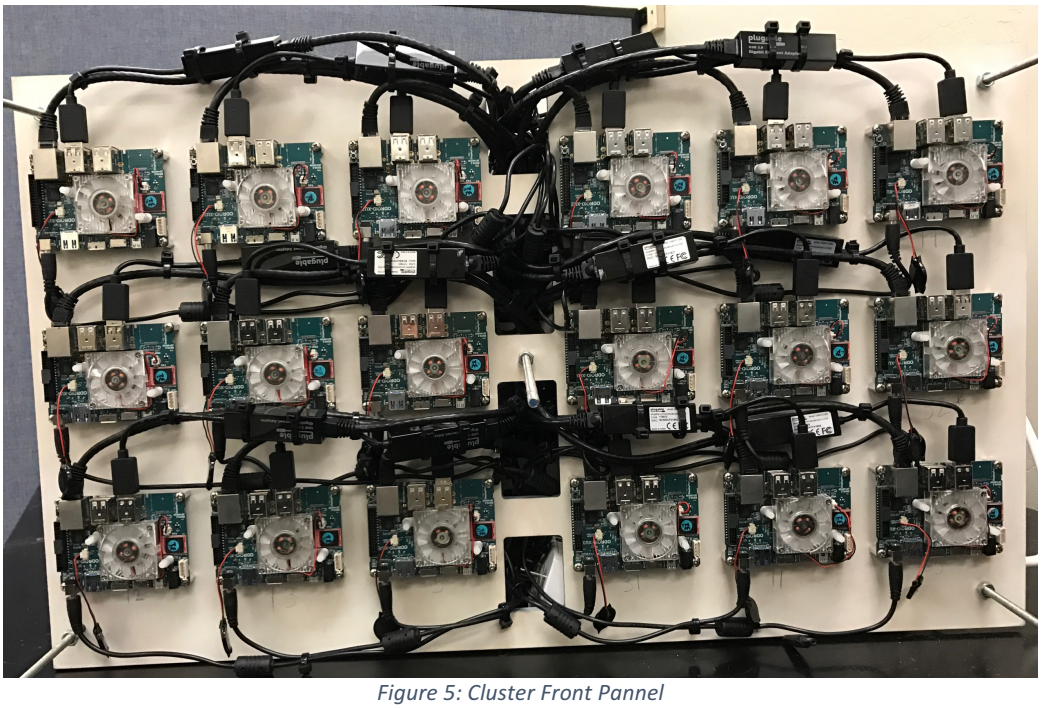

Figure 5: Cluster Front Pannel

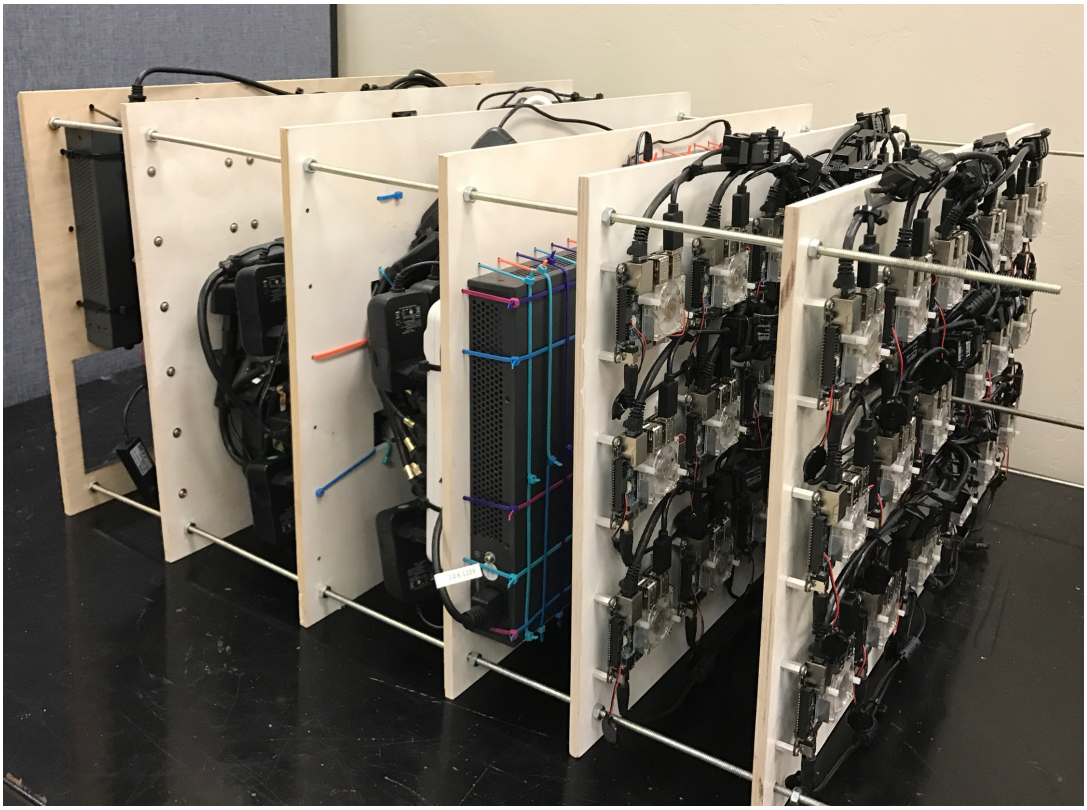

Figure 6: Overall Cluster Construction

Power Measurement Techniques

To analyze the overall consumption of both the cluster and the desktop computer, it was necessary to procure a commercial power meter. For this project, two separate power meters were used. Initially, a P3 International Kill-A-Watt was used to give instantaneous measurements of the total current draw from the cluster. This was done to ensure that the total electrical load did not exceed the allowable limit set by the circuit breaker of fifteen Amperes. As the goal of the project was to measure the overall consumption, it was necessary to have all the cluster computer boards, networking, and storage equipment connected to a single outlet so it could be measured in total. The tests conducted involved measuring the maximum current that could be drawn by all the individual components and summing them

together. At maximum CPU load for the computational nodes, the cluster would draw less than eight amperes. This showed that it was safe to connect the entire cluster through a single power meter.

After conducting the initial tests for current draw, it was necessary to obtain a new power meter that was capable of tracking and recording the power consumption over time. To accomplish this, an Elgato Eve Switch and Power Meter (see Figure 7) was obtained (Elgato). The Eve records the average power consumption over a span of ten minutes and averages those values to get the energy consumed in terms of Watt-hours. These results could be exported as a comma separated list that contained the time stamp and energy value.

Figure 7: Eve Energy Switch and Power Meter. Image from Elgato; Web; 10 May 2017 <https://www.elgato.com/sites/default/files/styles/desktop_hidpi_full/public/eve-energyus-solutionshot-icons-en.png?itok=-cmkPX8w&timestamp=1472205180>

Second Network

The Odroid boards used for this study have only one native Ethernet port. This port can

only handle data transfer of 100 Megabits per second (Mbps), which can be an issue in a cluster where the total amount of data being transferred is often several orders of magnitude larger. To counteract this, USB 3.0 to Gigabit Ethernet adapters were acquired and used to construct a second network that was rated at 1000 Mbps. This increase in speed allowed for faster data transfer between the nodes and the centralized storage as well as faster communications between the nodes themselves.

One issue that arose specifically with the Odroid XU3 was the USB 3.0 port. When the Ethernet dongle was connected to the USB 3.0 port, the software could recognize and communicate with the dongle. However, when large amounts of data, such as the kind of load that a cluster would typically undergo, was passed through the USB 3.0 port, the hardware controller for the port would overload and malfunction. This would cause the compute node to appear invisible on the faster network and the program would crash. The solution to this issue came in utilizing the USB 2.0's hardware controller and port at the cost of only being able to utilize half of the Gigabit Ethernet's bandwidth. This issue occurred in other instances and no notable solutions were found before the end of this study (Ubuntu Forums). From observations, the XU4 could process larger amounts of data without causing this issue, which is likely tied to the physical hardware of the XU3.

To quantify the value of the additional network, trials with identical initial conditions and number of nodes were run with only one network and then repeated when the second network was added. Doing this allowed to see if the cluster was being limited by the amount of communication bandwidth available as well as showing how much the efficiency of the cluster

is effect by the addition of more communicational throughput. With the additional network, the execution time decreased by 27.1% while the overall energy consumed was reduced by 16.1%. This suggests that the cluster is limited by the amount of network bandwidth that is available to pass the data between the nodes.

Desktop

For the purposes of comparison, a CFD workstation desktop was used to be able to compare against the cluster (see Figure 8). This desktop has and Intel i7 950 processor and 24 GB of RAM. The Intel i7 used here has four separate cores with hyper threading enabled that would allow for eight parallel processes to execute at the same time. The desktop chosen runs off a 64-bit operating system meaning that it can handle more points of data than the 32-bit operating system of the XU3 and XU4.

Figure 8: Desktop Computer

Software

To build a functioning cluster, it is necessary to tie all of the individual components of

the cluster together such that they work as a comprehensive unit. For a general layout, the cluster was built so that there was one node that acted as a head node and a management node at the same time. This node does not work on part of the computation, but rather acts as the communication point between all of the compute nodes and also works to manage all of the information that is being read and written to the external hard drive. All of the other nodes are called communication nodes and are carbon copies of each other except for their name and Internet Protocol (IP) address, which was supplied by two routers via Dynamic Host Configuration Protocol (DHCP). To make this layout effective, there are several key pieces of open source software that allow for the cluster to think that it was acting as one unit and not many individual computers.

Networked Filesystem

One of the more vital pieces of software was the Networked Filesystem (NFS). The most critical requirement for a cluster is to ensure that all the various computers are sharing the same information, namely the initial condition files. One method of accomplishing this would be to individually copy the data files from the compiling computer to each of the compute nodes MPI working directory. While this method does work, it vastly increases the possibility of error, corruption, or typos and takes a considerable amount of time from the user to copy the files. However, as the number of compute nodes grows increasingly in magnitude and often reaches into the thousands, this method becomes unfeasible. The other method is to utilize NFS. One particular node is chosen from the compute nodes and is designated as the master node. This node, often the node where the user compiles their program, takes one or more of

its folders and shares it to all the other nodes with a designated location or path and set of allowable actions or permissions that all the other nodes follow. Each of the other nodes can then mount that folder, just as a regular desktop mounts a USB flash drive, and read and write files depending upon the permissions specified. With NFS, however, every file that is written to is saved through the network onto the master node. This change is then seen by all the other nodes. This allows for the nodes to have access to all the necessary information all the time.

Network Time Protocol

Another key piece of software was the Network Time Protocol (NTP). Every time a node sends a packet of data, it records the exact time that it was sent based off its own reference for the current time stored in its local memory. When the other node receives the data, it will note the time that the file was sent to it. The issue, called clock skew, arises when the received time and sent time do not match. The data can become corrupt because the master node can receive data from what it believes to be the past and the future at the same time. This issue can be resolved by using one node that acts as a master time keeper and transmits this information to all the other nodes for reference. For this project, a dedicated communication node was connected to the internet through the university's network. The communication node used NTP to acquire the time from an external Ubuntu time server. The communication node then acted as a server that transmitted the data through the local cluster network to all the other compute nodes such that all the nodes in the cluster shared the same timestamp (Microsemi).

OpenMPI

The most critical piece of software for the cluster was the Message Pathing Interface, or

MPI. MPI is a software package that splits the high-level computations defined by an executable file into smaller sets. These smaller sets are distributed to each of the compute nodes. In addition, MPI also contains a set of instructions for communicating computational information between nodes. MPI is what takes a serial problem and transforms it into a parallel one. There are two main implementations of MPI in the computing world. MPICH is the most commonly known MPI program and is most often used for smaller cluster implementations because of its ease of use. However, for most scientific computing, an MPI application called OpenMPI is used. Both MPICH and OpenMPI are open source software, meaning that everything about it can be accessed and changed by anyone with relevant technical knowledge. This allows for individual clusters to optimize the message pathing to be fully optimized for their architecture and network layout.

Ubuntu Operating System

Like most computers, the Odroid boards come preinstalled with a particular release of an operating system. For the XU3 and XU4, that is the Ubuntu 14.04 operating system. 14.04 was originally released in 2014 and was at the end of its lifecycle when this project was started. Midway through the project, the operating system was updated to the more current 16.04 release because many of the individual components internal to the software were outdated. The main component updated with the newer operating system was the kernel. For all computers, it is necessary to have a supporting buffer between the hardware and the software and that is the role of the kernel. HardKernel custom makes its own kernel because it produces its own hardware. With that, only certain iterations of larger software releases are actually

supported on the XU3 and XU4. When the newer operating system and kernel were installed, many incompatibilities that arose under the 14.04 were resolved instantly.

Results

Methodology

The overall purpose of the study relied on directly relating the amount of overall energy consumed and instantaneous power consumption to the amount of computational resources that were available regardless of which platform the computation is run on. The two platforms used for this study were the constructed Beowulf cluster and desktop computer described above. To remove all possible variables, a Fortran based Computational Fluid dynamics problem was used for the computation. The code can either generate a random set of initial conditions or use a precompiled set of data. To remove the variability, a precompiled set of initial conditions was used such that the amount of time and energy required to complete the computations was purely based on the overall computational throughput available. The amount of energy consumed was taken with the sum total of all necessary components to run the computation. These components included all of the individual compute nodes, switches, routers, monitors, and the external hard drive.

To study the computational efficiency, the number of parallel processes was varied to understand its effect on the overall execution time and power consumption. For the desktop, this could only be varied by adding additional processes to the individual cores of the processors. For the cluster, this process became more complicated because of the two-fold limiting factor of the processors and the communication. Increasing the overall number of

processes assigned to all of the nodes allowed for the entirety of the networks bandwidth to be used. Changing the number of processes assigned to each individual node allowed for the total bandwidth of the individual boards Ethernet interfaces to be utilized. The number of processes per node also affects how much of the individual processors are used.

Overall Results

Table 1 shows the results gained from changing the variables discussed above in terms of the power required to operate, the total time to execute, and the overall energy that was required to complete the computations. The total number of parallel executions can be obtained by multiplying 'Number of Nodes' by 'Number of Processes Per Node' for any given row. The 'Energy Consumed (kWh)' column represent to total amount of energy consumed and it the product of the 'Power Consumed (W)' and 'Time Elapsed (min)' following correct unit conversions. The 'Total Cost' column represents the cost an average San Diego homeowner would pay for that particular run.

Number of Nodes	Number of Processes Per Node	Power Consumed (W)	Energy Consumed (kWh)	Time Elapsed (min)	Total Cost
Cluster Results					
4	1	100.43	4.598	2,747	$0.97
8	1	95.00	3.2395	2,046	$0.68
16	1	164.73	2.60549	949	$0.55
16	1	162.89	2.57909	950	$0.54
16	2	187.76	2.48157	793	$0.52
32	1	258.02	2.50712	583	$0.53
32	2	310.01	3.653	707	$0.77
32	4	365.75	5.79712	951	$1.22
Desktop Results					
1	1	127.73	1.13893	535	$0.24
1	2	140.71	0.67777	289	$0.14
1	4	178.31	0.58842	198	$0.12

Table 1: Overall Results

To see the effect of the communication, the number of nodes used for the computation was steadily increased. When the speed up of increasing the number of nodes no longer became significant, the number of processes per node was then increased to maximized the overall bandwidth used. The run that consumed the least amount of energy occurred when there were sixteen nodes with two processes per node and the fastest run occurred when there were thirty-two nodes with one process each. Both trials were significant because they showed that the overall computation was being limited by the amount of communication speed and bandwidth that was available. This can be confirmed by looking at the trial with thirty-two nodes and two processes per node. Increasing the overall amount of parallel tasks did not decrease the overall power consumed suggesting that the processors were not being maximized. One trial that is interesting to note is with thirty-two nodes and four processes per

node. With four processes per node, the processors were consuming enough energy such that they were triggering the attached cooling fans. This created a significant spike in the overall energy consumed as none of the other trials dissipated enough heat to trigger the fans.

To compare this result to the desktop computer, a graph was created showing the average energy consumption over a ten-minute interval for the duration of the computation (see Figure 9). In total, the desktop computer was much faster and consumed much less energy to complete the computations. By comparing the slowest run of the desktop using a serial computational structure and the fastest run of the cluster, it becomes apparent the differences between the two resources. At its best, the cluster consumes twice the amount of energy compared to the desktop running at its slowest. This conclusion shows the overall trend of the data but does not show the efficiency or any other metrics of the data.

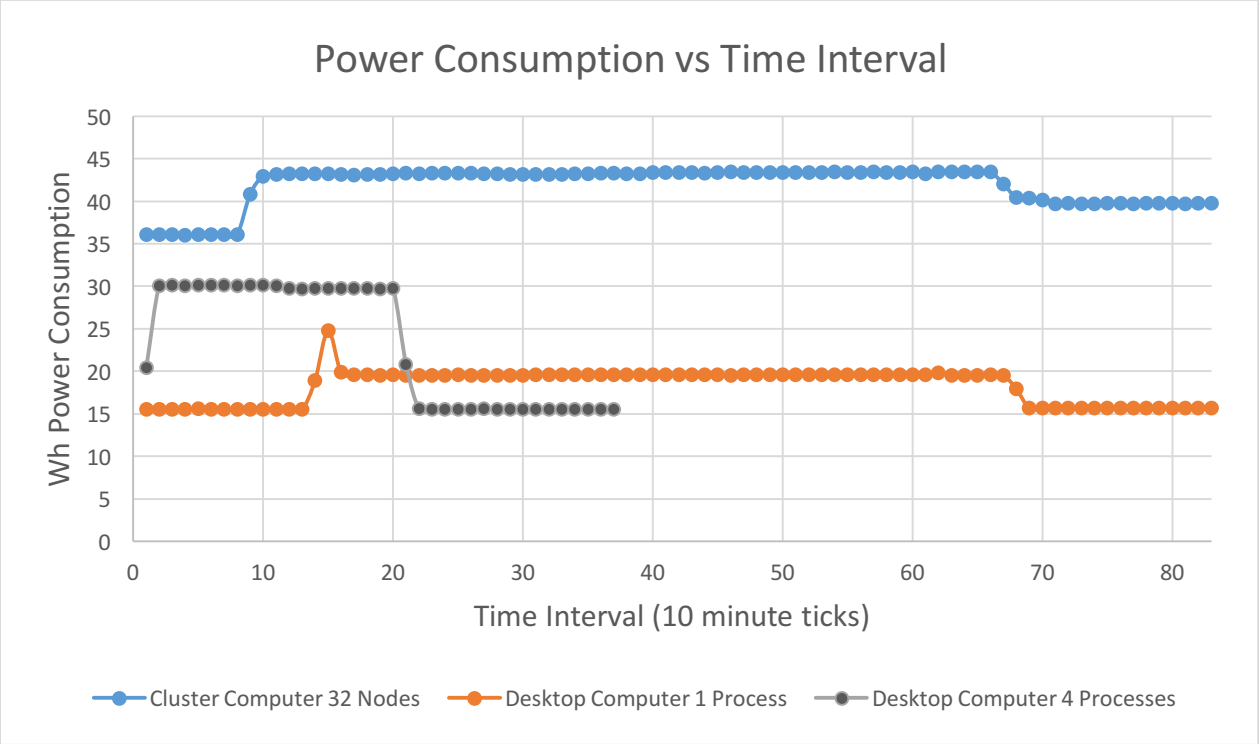

Figure 9: Power Consumption vs Time Interval

Idle Results

To further analyze how the data compares, the power draw during an idle state was taken based from the number of nodes that were active but in an idle state. Table 2 shows the results of this analysis.

Number of Nodes	Power (W)	Energy (kWh)	Time Elapsed (min)
Desktop Computer			
1	93.88	1.97	1260
Cluster Computer			
4	94.62	2.11	1340
8	102.92	13.23	7715
16	123.62	4.64	2252
32	236.27	22.69	5763

Table 2: Idle Results

The most interesting result is the amount of power consumption that the cluster consumes. For the fastest trial of the cluster, 91.6% of the energy is just from the idle based operations. This is a very interesting result because it is significantly higher than the 61.2% found in the study done at Boise State (Kiepert). To show the effect of this, the average idle consumption for each data set was subtracted from its corresponding instance in Figure 9 (see Figure 10). When comparing the overall amount of energy consumed purely for computations, the cluster performs on the same level of the desktop.

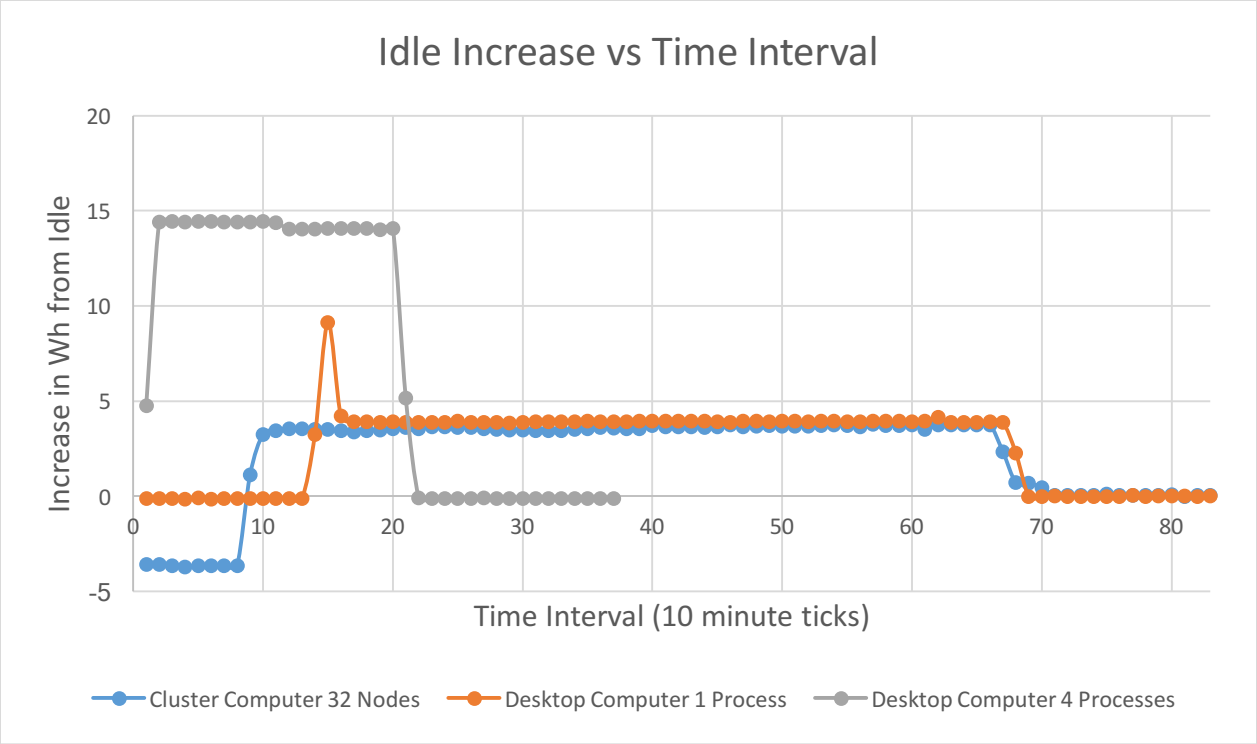

Figure 10: Idle Increase vs Time Interval

LINPACK Benchmark

To test the limiting factor of the cluster, it was necessary to perform a metric that quantifies the computational power of an individual node inside the cluster. To do this, a HPL benchmark was performed, which is a portable implementation of the LINPACK benchmark (Petitet, Cleary and Dongarra). The LINKPACK benchmark is a subset of Fortran based codes used to solve linear algebra problems (Dongarra). The HPL benchmark calculates the number of FLOPS that the individual node can perform, which is the same benchmark performed by the TOP500 supercomputers. Theoretically, the maximum performance of the cluster is the sum of all of the individual nodes used to make it up. To this effect, the theoretical output of the cluster is around 50 MFLOPS and the desktop is close to 5 MFLOPS. The cluster should be on the order of ten times the computations performance of the desktop. This suggests that the

overall limiting factor of the cluster is in the speed and amount of communication bandwidth available. This finding shows that the overall computational results are skewed because the overall peak performance of the cluster is not being used.

Power Efficiency

The other data metric that can be used to analyze the individual performance comparison is in terms of the power efficiency. For this study, the power efficiency was looked at in terms of the overall energy consumption and the average instantaneous power. This metric can be used to show how well the desktop and cluster scale with an increasing number of processes. Equation 2 was used to evaluate the power efficiency where the numerator represents the trial with the least number of parallel executions and the denominator is the parallel case with n number of processes. For the average instantaneous power efficiency (see Figure 11), the overall trends of the data between the cluster and desktop follow the same trend. This trend is also followed when the power efficiency in terms of overall energy is graphed (see Figure 12).

In an ideal case, the amount of total resources consumed in a serial case would be the same or less than the amount consumed in a parallel case times the number of parallel executions. The power efficiency equation shows how close the experimental results are to this theoretical maximum. The graphed trend lines of the cluster and desktop follow similar paths but are shifted by a given amount. This suggests that the internal processes of executing the parallel structures are similar, but there is an initial difference that causes all the desktop executions to be more efficient.

$$E = \frac{n_{ref} x_{ref}}{n_i x_i}$$

Equation 2: Power Efficiency

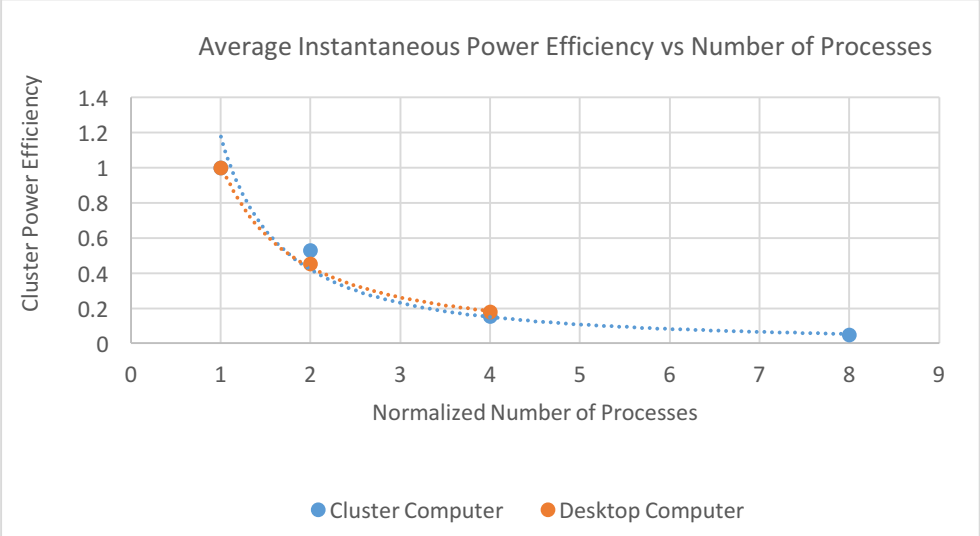

Figure 11: Average Instantaneous Power Efficiency vs Number of Processes

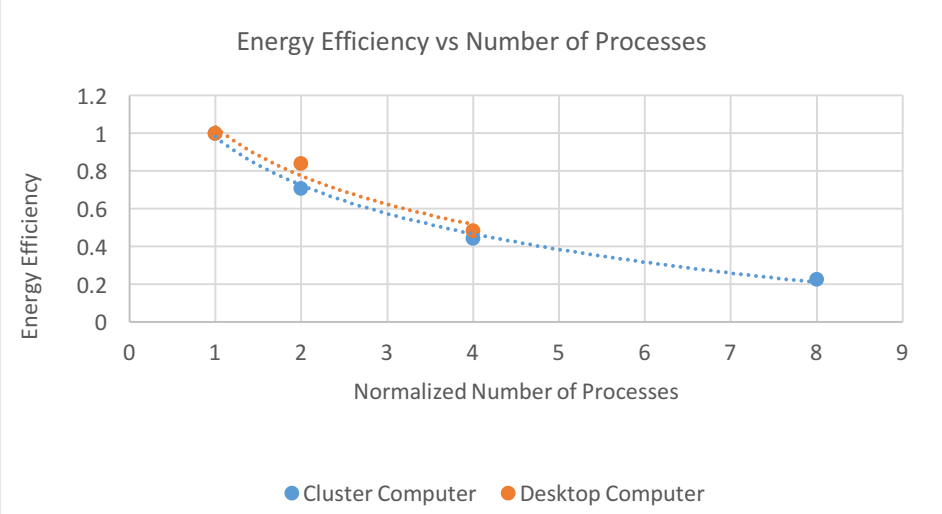

Figure 12: Energy Efficiency vs Number of Processors

Conclusions

The results of this study show that the current feasibility of using ARM based processors for use in low power High Performance Clusters is limited. The limitations imposed by the use

of a Beowulf cluster limit the overall ability to use high grade networking equipment to achieve the maximum theoretical computational ability of the cluster. This limitation is that the cost of the cluster would then significantly increase, which defeats the definition of a COTS Beowulf clusters. In addition, the amount of power used by the cluster sitting idly in comparison to a running state shows that there is a lot of inefficiencies in how the processors manages key computational tasks.

Moving forward, there are several key areas that can be addressed to improve the cluster. The first area is in the Odroid boards themselves. Most of the cluster was comprised of XU3 based boards, which were released in 2014 (HardKernel). The newer XU4 based boards have improved hardware including an updated USB3.0 Ethernet controller and a native Gigabit Ethernet connection. These two updates would significantly increase the available speed of the communication, which would greatly improve the overall performance of the cluster without increasing the rate that power is consumed. The other variable that can be studied is the individual power regulators. Currently, each computational board has its own regulator that supplies the board with 5 volts. Inherent with each of these regulators is some amount of power that is lost. Summed across all of the individual regulators, the loss can have a substantial effect on the overall readings taken at the wall. To study the effect of this, a bused power system with one regulator would show how much power is being lost to the individual regulators.

Works Cited

Balakrishnan, Nikilesh. "Building and Benchmarking a Low Power ARM Cluster." 24 August 2012. *The University of Edinburgh.* January 2017. <https://static.ph.ed.ac.uk/dissertations/hpc-msc/2011-2012/Submission-1126390.pdf>.

Buytaert, Kris. *Chapter 2. So what is openMosix Anyway*. 18 June 2004. 20 April 2017. <http://www.tldp.org/HOWTO/openMosix-HOWTO/x135.html>.

Dongarra, Jack. *Frequently Asked Questions on the Linpack Benchmark and Top500*. 8 May 2007. 22 May 2017. <http://www.netlib.org/utk/people/JackDongarra/faq-linpack.html>.

Elgato. *eve energy Switch & Power Meter*. n.d. 22 May 2017. <https://www.elgato.com/en/eve/eve-energy>.

ENERGY STAR. "ENERGY STAR Market & Industry Scoping Report Coffee Makers." November 2011. 12 April 2017. <https://www.energystar.gov/sites/default/files/asset/document/ENERGY_STAR_Scoping_Report_Coffee_Makers.pdf>.

Greene, Kate. *MIT Technology Review*. 1 June 2006. 15 March 2017. <https://www.technologyreview.com/s/405892/intels-new-strategy-power-efficiency/>.

HardKernel co., Ltd. *Odroid*. n.d. 22 May 2017. <http://www.hardkernel.com/main/main.php>.

HardKernel. *Odroid-XU3 Lite*. n.d. 10 May 2017. <http://www.hardkernel.com/main/products/prdt_info.php?g_code=G141351880955>.

Kamil, Shoaib, Joan Shalf and Erich Storhmaier. *Power Efficiency in High Performance Computing*. n.d. 22 May 2017. <https://crd.lbl.gov/assets/pubs_presos/CDS/ATG/powereffreportfull.pdf>.

Kiepert, Joshua. *Creating a Raspberry Pi-Based Beowulf Cluster*. 22 May 2013. 21 April 2017. <http://coen.boisestate.edu/ece/files/2013/05/Creating.a.Raspberry.Pi-Based.Beowulf.Cluster_v2.pdf>.

Levy, Markus. *he History of The ARM Architecture: From Inception to IPO*. n.d. 1 May 2017. <http://reds.heig-vd.ch/share/cours/ReCo/documents/TheHistoryOfTheArmArchitecture.pdf>.

Meuer, Hans, et al. "NOVEMBER 2016." November 2016. *TOP500.* 12 April 2017. <https://www.top500.org/lists/2016/11/>.

Microsemi. *Clock Skew and Short Paths Timing*. June 2011. 10 May 2017. <https://www.microsemi.com/document-portal/doc_view/129896-ac198-clock-skew-and-short-paths-timing-app-note>.

Petitet, A., et al. *HPL - A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers*. n.d. 22 May 2017. <http://www.netlib.org/benchmark/hpl/>.

Raspberry Pi Foundation. n.d. 22 May 2017. <https://www.raspberrypi.org>.

Raspberry Pi. *Raspberry Pi 2 Model B*. n.d. 10 May 2017. <https://www.raspberrypi.org/products/raspberry-pi-2-model-b/>.

San Diego Supercomputer Center. *30 Years of Turning Data to Discovery*. n.d. 22 May 2017. <https://www.sdsc.edu/about_sdsc/sdsc_timeline_web.pdf>.

Scogland, Tom and Wu-chen Feng. *Green500*. November 2016. 20 April 2017. <https://www.top500.org/green500/lists/2016/11/>.

TOP500. *Introductions and Objectives*. n.d. 22 May 2017. <https://www.top500.org/project/introduction/>.

U.S. Energy Information Administration. *How much electricity does an American home use?* 18 October 2016. 22 May 2017. <https://www.eia.gov/tools/faqs/faq.php?id=97&t=3>.

Ubuntu Forums. *0b95:1790 [Asus N55SF] Bad performance of Asix Ethernet-to-USB device on USB3 port*. 16 January 2014. 8 December 2016. <https://bugs.launchpad.net/ubuntu/+source/linux/+bug/1269883>.